窗口模块设计文档

# 窗口结构







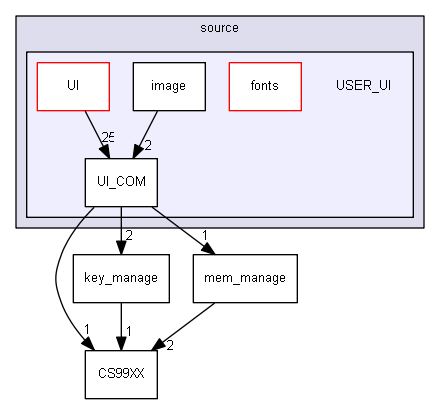






# 目录依赖关系图

## USER\_UI的目录依赖关系图:

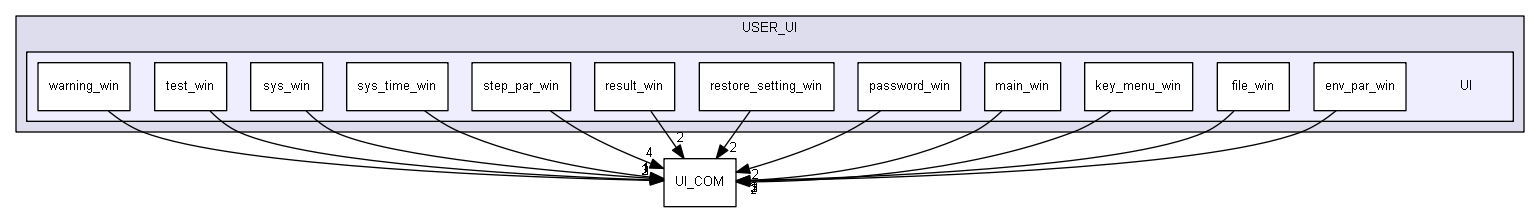


key\_manage是按键管理模块

mem\_maage是存储管理模块

CS99XX是上层系统数据结构模块

## UI目录依赖关系图



warning\_win警告窗口，此窗口是提供给其他窗口来使用的

test\_win测试窗口

sys\_win系统管理窗口

sys\_time\_win系统时间管理窗口

step\_par\_win步骤参数管理窗口

result\_win结果管理窗口

restore\_setting\_win恢复出厂设置窗口

password\_win密码管理窗口

main\_win主窗口

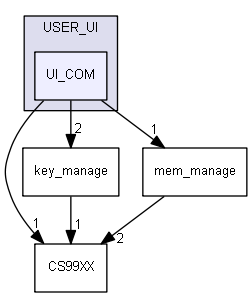
key\_menu\_win按键管理窗口

file\_win文件管理窗口

env\_par\_win环境参数管理窗口

所有以上窗口均对UI\_COM产生依赖，窗口的创建接口均向上注册到UI\_COM中(接口声名);

## UI\_COM的目录依赖关系图



注:USER\_UI是所有界面的存放目录

## SER\_UI的目录

UI\_COM窗口公共文件目录

UI窗口文件目录

fonts字体目录

image图片目录

UI\_COM目录文件

com\_ui\_info.c 窗口公共接口定义

com\_ui\_info.h 窗口公共接口声名

com\_edit\_api\_c 编辑控件操作的接口定义

com\_edit\_api.h 编辑控件操作的接口声名

key\_server.c 按键服务函数接口定义，按键服务任务

key\_server.h 按键服务任务对外提供的接口声名

UI目录下子目录

env\_par\_win 环境参数窗口

file\_win 文件窗口

key\_menu\_win 按键菜单窗口

main\_win 主窗口

password\_win 密码管理窗口

restore\_setting\_win 恢复出厂设置窗口

result\_win 结果管理窗口

self\_check\_win 自检窗口

start\_win 启动窗口

step\_par\_win 步骤参数管理窗口

sys\_time\_win 系统时间设置窗口

sys\_win 系统管理窗口

test\_win 测试窗口

warning\_win 警告提示窗口

界面模块依赖两个任务：1.emwin管理任务2.按键服务任务

## 1.emwin管理任务在main\_win文件中的main\_ui\_enter函数





## 2.按键服务任务



界面模块建立在三大结构体之上：

1. 用户窗口

MYUSER\_WINDOW\_T

1. 文本对象

TEXT\_ELE\_T

1. 编辑对象

EDIT\_ELE\_T

用户窗口结构说明

struct MYUSER\_WINDOW{

uint8\_t \*win\_name[LANGUAGE\_NUM];///<窗口名称

USER\_CALLBACK call\_back\_fun;///<回调函数

void (\*init\_key\_fun)(WM\_HMEM);///<窗口配套的按键更新函数

ELE\_POOL\_INF text;///<文本控件索引池

ELE\_POOL\_INF edit;///<编辑控件索引池

ELE\_POOL\_INF com;///<公共文本控件索引池

AUTO\_LAYOUT\_POOL auto\_layout;///<自动布局信息

WIDGET\_POS\_SIZE\_T \*\*pos\_size\_pool;///<窗口的位置尺寸池

WIDGET\_POS\_SIZE\_T pos\_size;///<窗口的位置尺寸

WM\_HMEM handle;///< 窗口句柄

CS\_LIST w\_list;///< 窗口链表

};

成员说明：

win\_name 窗口名称字符串数组，数组长度是系统所支持的语言个数 LANGUAGE\_NUM

call\_back\_fun 窗口回调函数，由用户实现回调函数并用回调函数来初始化这个成员变量

init\_key\_fun 初始化按键信息的函数，由用户针对不同的窗口来实现各自的按键初始化函数

test 窗口中所有的私有文本控件信息池;如果为空表示窗口中没有私有的文本对象

edit 窗口中所有的私有编辑控件信息池;如果为空表示窗口中没有私有的编辑对象

com 窗口中所有的公共文本控件信息池;如果为空表示窗口中没有用到公共的文本对象

auto\_layout 自动布局结构信息;这个成员可以为空NULL，表示不使用自动布局

pos\_size\_pool 窗口的位置尺寸信息池;针对不同尺寸的屏幕特意设计的;如果为空表示程序员要手动对pos\_size成员进行初始化

pos\_size 窗口的位置尺寸信息;可能的初始化途径有：

1.当提供了pos\_size\_pool信息后，程序会自动的进行初始化;

2.当pos\_size\_pool为空时表示程序在其他地方人为进行了初始化操作;

handle 窗口句柄;此句柄在窗口被创建时，由窗口回调函数进行初始化;当窗口被删除时窗口句柄会被清0;

文本结构说明

typedef struct TEST\_UI\_ELE{

uint8\_t \*text[LANGUAGE\_NUM];///< 内容

CS\_INDEX index;///<索引

UI\_ELE\_DISPLAY\_INFO\_T dis\_info;///<显示信息

WM\_HMEM handle;///< 句柄

CS\_LIST list;///< 文本链表

}TEXT\_ELE\_T;

成员说明:

text 文本内容数组;数组元素个数由系统所支持的语言个数来决定;

index 文本对象在文本对象池中的索引值;设计索引值的目的是要实现窗口中对象的查找，以及初始化配置文本对象表，只有在配置表中有的文本对象索引值，才会被创建出来;

dis\_info 显示信息结构用来控制文本对象的所有显示相关信息

typedef struct UI\_ELE\_DISPLAY\_INFO{

uint16\_t base\_x;///< x基坐标(像素)

uint16\_t base\_y;///<y基坐标(像素)

WIDGET\_POS\_SIZE\_T pos\_size;///<窗口的位置尺寸

uint8\_t max\_len;///< 最大长度

const GUI\_FONT \* font;//字体

GUI\_COLOR font\_color;///<字体颜色

GUI\_COLOR back\_color;///<背景颜色

int align;///< 对齐方式

}UI\_ELE\_DISPLAY\_INFO\_T;

结构成员说明

base\_x x基坐标;文本对象采用了基坐标+偏移坐标的方法来进行配置

base\_y y基坐标文本对象采用了基坐标+偏移坐标的方法来进行配置

pos\_size 文本对象的位置尺寸信息

位置尺寸信息结构定义

typedef struct{

uint16\_t x;///<x坐标(像素)

uint16\_t y;///<y坐标(像素)

uint16\_t width;///<宽(像素)

uint16\_t height;///<高(像素)

}WIDGET\_POS\_SIZE\_T;

x x相对坐标位置

y y相对坐标位置

width 宽度

height 高度

max\_len 文本的最大长度

font 字体

font\_color 字体颜色

back\_color 背景颜色

align 对齐方式

handle 文本控件句柄当文本被创建时被初始化，当文本对象被删除时文本对象句柄会被清0;

list 文本对象的链表;窗口中所有的文本对象都是通过链表来维护管理的;

编辑对象结构

struct EDIT\_ELE\_T\_{

uint8\_t \*name[LANGUAGE\_NUM];///< 名称

CS\_INDEX index;///< 通过枚举索引

uint8\_t \*value[10];///< 默认值

/\* 数据 \*/

struct{

void \*data;///< 数据指针

uint8\_t bytes;///< 字节数

}data;

/\* 资源 \*/

struct{

void \*table;///< 资源表

uint8\_t size;///< 资源表的条数

void \*user\_data;///<用户数据

uint8\_t user\_data\_size;///<用户数据个数

void \*table\_2[LANGUAGE\_NUM];///< 可包含不同语言的资源

}resource;

/\* 类型 \*/

struct{

ELE\_TYPE type;///< 对象类型

}type;

/\* 格式 \*/

struct{

uint8\_t dec;///< 小数点位数

uint8\_t lon;///< 数据显示长度,如果是浮点数包含小数点，必须小于10

UNIT\_T unit;///< 单位

}format;

/\* 范围 \*/

struct{

uint32\_t high;///< 上限

uint32\_t low;///< 下限

uint8\_t \*notice[LANGUAGE\_NUM];///< 提示信息包含中英文

void (\*check\_value\_validity)(EDIT\_ELE\_T\*,uint32\_t\*);///<检查数据的正确性

void (\*provided\_dis\_range\_fun)(EDIT\_ELE\_T\*);///<提供的显示range的函数

}range;

/\* 按键信息 包含系统键 菜单键 键盘服务函数 \*/

struct{

void (\*fun\_sys\_key)(WM\_HMEM);///< 方向键信息更新函数

void (\*fun\_menu\_key)(WM\_HMEM);//菜单键更新函数

void (\*fun\_key)(uint32\_t);///< 设置该变量的键盘服务函数

}key\_inf;

EDIT\_ELE\_DISPLAY\_INF dis;///<显示相关的配置信息

uint8\_t num;///<显示页中的编号0-n

uint8\_t total;///<所在页总共的编辑控件个数

uint8\_t page;///<所在的页编号1-n

CS\_LIST e\_list;///< 编辑控件链表

};

编辑对象结构是一个比较复杂的结构：



一个编辑对象由三部分组成：编辑对象名称(文本)、编辑控件(可编辑控件编辑框、下拉框)、单位(文本)

name 编辑对象的名称数组;元素个数由系统所支持的语言个数决定;

index 枚举索引通过索引值可以从编辑对象池中查找到编辑对象的实体;同时在窗口编辑对象配置表中出现的索引值其编辑对象才会被创建出来;

value 编辑控件的默认值;基本无用

data 编辑对象所邦定的变量数据信息

resource 资源信息信息 当编辑控件是一个下拉框时会用到

table 资源表指针;通常被初始化为指向一个一维字符串数组如const uint8\_t \*work\_mode\_pool[2]={"N","G"};

size 表示资源表中资源个数;如上例中work\_mode\_poo 资源个数是2可以用ARRAY\_SIZE(work\_mode\_poo)计算出

user\_data 用户数据表设计它是为了解决下拉框中的第一个元素表示的可能与在下拉标签顺序不对应;

例：一个仪器型号被配置为DCW,IR两个测试模式,table指向的资源表是{“DCW”,”IR”};那么DCW是下拉框的第一个标签，IR是下拉框的第二个标签;在程序中DCW表示的数值是2，IR是3;为了使DCW与2关联，IR与3关联,设计了user\_data指向一个数组flag[2]={2,3};可以这样解释;通过size成员变量知道了测试模式的个数，通过下标0-size-1依次访问所有的资源表信息(table中)和对应的数值信息在user\_data中;

user\_data\_size 用户数值信息表中元素的个数，通常等于size成员变量

table\_2 可包含不同语言的资源表数组，数组元素个数由系统支持的语言个数决定;类比table的使用过程发现多是与系统语言无关如”N”,”G”,”DCW”,”IR”全是英文字符;当出现有的下拉框根据不同的系统语言会有中英文两种字符时需要使用这个资源表。

例如：在设置一些开关型变量时用到const uint8\_t \*sw\_str[2][2]={{“开启”,”关闭”},{“ON”,”OFF”}};

注：table和table\_2到底那个起作用？当table和table\_2中不为空的那个起作用，如果两个都不为空，那么tabl\_2起作用

type 类型标明编辑对象是编辑框或下拉框

type 文本类(不再使用)、数字编辑框类、字符串编辑框类、下拉框类

ELE\_TEXT,///< 文本

ELE\_EDIT\_NUM,///< 数字编辑框

ELE\_EDIT\_STR,///< 字符串编辑框

ELE\_DROPDOWN,///< 下拉选择框

format编辑对象的显示格式，针对数字编辑框专门设计的如设置电压时格式为:5.000kV

dec 小数点个数;例：5.000kV其dec值为3

lon 总长度包含小数点;例：5.000kV其lon值为5

unit 单位索引值;例:5.000kV其unit值为VOL\_U\_kV

单位的索引枚举如下:

typedef enum{

NULL\_U\_NULL,///<无单位

TIM\_U\_s,///<秒

VOL\_U\_mV,///<mV

VOL\_U\_V,///< V

VOL\_U\_kV,///< kV

CUR\_U\_uA,///< uA

CUR\_U\_mA,///< mA

CUR\_U\_A,///< A

RES\_U\_mOHM,///< mOhm

RES\_U\_OHM,///< ohm

RES\_U\_MOHM,///< Mohm

RES\_U\_GOHM,///< Gohm

CAP\_U\_pF,///< pF

CAP\_U\_nF,///< nF

CAP\_U\_uF,///< uF

FREQ\_U\_Hz,///< Hz

U\_PER\_CENT,///< (%)

}UNIT\_T;

range 参数的范围管理

high 参数上限值

low 参数下限值

notice 提示信息的字符串数组，数组元素个数由系统所支持的语言个数决定;

check\_value\_validity 函数指针;这个函数指针有两种用途。对于普通参数不需要进行特殊处理的就将这个函数指针赋空。

1. 是为那些不能用简单的上下限来描述其范围的参数如（测试时间范围是0，0.3-999.9s）.
2. 设置这个参数与其他参数相关联的要进行调整。如设置下限值时，他与上限值相关联，设置上限时又会影响到下限的设置值。

provided\_dis\_range\_fun 函数指针是提供给一些特殊需求下来显示参数的范围提示信息的;

例如:测试时间的设置他就比较特殊。格式是0,0.3-999.9s;就需要提供一个这样的函数来专门处理。

对于那些普通的使用上下限就可以描述清楚的参数就不使用这个函数指针就赋空

key\_inf 提供键盘信息的结构;设置参数时用需要键盘进行配合。当编辑控件被选为当前的编辑对象时就会用这个结构信息进行按键的初始化

fun\_sys\_key 系统按键信息初始化函数指针;包含：码盘左、右，上、下、左、右键，U/LOCK,OFFSET,ENTER,EXIT,SHIFT键和组合按键如F1+KEY\_0,F1+KEY\_1等

fun\_menu\_key 菜单键信息初始化函数指针;针对菜单键F0,F1,F2,F3,F4,F5,F6

fun\_key 数字键盘的扫描函数初始化指针;针对不同的输入场景进行配置：

如设置文件名时用：keyboard\_fun\_str

设置数值型参数时用：keyboard\_fun\_num

设置密码时用: keyboard\_fun\_pwd

设置测试端口时用: keyboard\_test\_port

dis 与显示相关的配置信息

显示相关结构定义

typedef struct{

CONFIGURABLE\_PROPERTY name;///< 名称

CONFIGURABLE\_PROPERTY edit;///< 编辑

CONFIGURABLE\_PROPERTY unit;///< 单位

uint16\_t x;///< 所在窗口的 x 坐标

uint16\_t y;///< 所在窗口的 x 坐标

}EDIT\_ELE\_DISPLAY\_INF;

name 名称的显示配置信息

edit 编辑控件的显示配置信息

uint 单位的显示配置信息

x 所在窗口的横坐标（像素）

y 所在窗口的纵坐标(像素)

配置信息结构定义

typedef struct{

uint16\_t width;///<宽

uint16\_t height;///<高

const GUI\_FONT \*font;///<字体

GUI\_COLOR font\_color;///<字体颜色

GUI\_COLOR back\_color;///<背景颜色

int align;///<对齐方式

uint32\_t max\_len;///<最大长度

WM\_HMEM handle;///<句柄

}CONFIGURABLE\_PROPERTY;

width 控件宽度(像素)

height 控件高度(像素)

font 字体

font\_color 字体颜色

back\_color 背景颜色

align 对齐方式

max\_len 最大长度

heandle 控件句柄

num 在当前页中的编号0-n

total 当前页中总共用的编辑对象个数

page 所在的页编号1-n

e\_list 编辑对象链表;本界面模块中所有的编辑控件都是通过链表进行维护管理的。







